

What is claimed is:

1 1. A system for detecting termination of an application instance using
2 locks, comprising:
3 a holding child process forked from a parent process, the holding child
4 process comprising a connection to a monitored application instance and an
5 exclusive lock on the monitored application instance, the holding child process
6 returning a ready signal upon successfully acquiring the exclusive lock;
7 a waiting child process forked from the parent process subsequent to the
8 holding child process, the waiting child process comprising a connection to the
9 monitored application instance, the waiting child process blocking on the
10 exclusive lock on the monitored application instance and returning a result signal
11 upon at least one of acquiring the exclusive lock and clearing the block on the
12 exclusive lock; and
13 the parent process processing the return signal.

1 2. A system according to Claim 1, further comprising:
2 the parent process processing a standard error received from the waiting
3 child process.

1 3. A system according to Claim 1, further comprising:
2 the parent process processing a non-standard error received from the
3 waiting child process.

1 4. A system according to Claim 3, further comprising:
2 a validation module checking for termination of the monitored application
3 and signaling termination of the monitored application to a cluster service.

1 5. A system according to Claim 3, further comprising:
2 a validation module checking for termination of the monitored application
3 and restarting the holding child process and the waiting child process.

1 6. A system according to Claim 1, wherein the application instance
2 comprises a database server instance.

1 7. A method for detecting termination of an application instance
2 using locks, comprising:
3 starting a holding child process from a parent process, comprising:
4 connecting to a monitored application instance;
5 acquiring an exclusive lock on the monitored application instance;
6 and
7 returning a ready signal upon successfully acquiring the exclusive
8 lock; and
9 starting a waiting child process from the parent process subsequent to the
10 holding child process, comprising:
11 connecting to the monitored application instance;
12 blocking on the exclusive lock on the monitored application
13 instance; and
14 returning a result signal upon at least one of acquiring the
15 exclusive lock and clearing the block on the exclusive lock; and
16 processing the return signal at the parent process.

1 8. A method according to Claim 7, further comprising:
2 processing a standard error received from the waiting child process.

1 9. A method according to Claim 7, further comprising:
2 processing a non-standard error received from the waiting child process.

1 10. A method according to Claim 9, further comprising:
2 checking for termination of the monitored application; and
3 signaling termination of the monitored application to a cluster service.

1 11. A method according to Claim 9, further comprising:
2 checking for termination of the monitored application; and
3 restarting the holding child process and the waiting child process.

1 12. A method according to Claim 7, wherein the application instance
2 comprises a database server instance.

1 13. A computer-readable storage medium holding code for performing
2 the method according to Claim 7.

1 14. A system for detecting termination of a database instance using
2 events, comprising:
3 a waiting subroutine forked from a main routine, the waiting subroutine
4 comprising a connection to a monitored database instance and blocking on a
5 named event in the database instance and returning a result to the main routine
6 upon an occurrence of the named event; and
7 the main routine processing the result.

1 15. A system according to Claim 14, wherein the named event
2 comprises a membership change event, further comprising:
3 the main routine processing the membership change event and restarting
4 the waiting subroutine.

1 16. A system according to Claim 14, further comprising:
2 the main routine processing a standard error received from the waiting
3 subroutine.

1 17. A system according to Claim 14, further comprising:
2 the main routine processing a non-standard error received from the waiting
3 subroutine.

1 18. A system according to Claim 17, further comprising:
2 a validation module checking for termination of the monitored named
3 instance application and signaling termination of the monitored named instance
4 application to a cluster service.

1 19. A system according to Claim 17, further comprising:
2 a validation module checking for termination of the monitored named
3 instance application and restarting the waiting subroutine.

1 20. A method for detecting termination of a database instance using
2 events, comprising:

3 starting a waiting subroutine from a main routine, comprising:
4 connecting to a monitored database instance;
5 blocking on a named event in the database instance; and
6 returning a result to the main routine upon an occurrence of the
7 named event; and
8 processing the result at the main routine.

1 21. A method according to Claim 20, wherein the named event
2 comprises a membership change event, further comprising:

3 processing the membership change event at the main routine; and
4 restarting the waiting subroutine.

1 22. A method according to Claim 20, further comprising:
2 processing a standard error received from the waiting subroutine.

1 23. A method according to Claim 20, further comprising:
2 processing a non-standard error received from the waiting subroutine.

1 24. A method according to Claim 23, further comprising:
2 checking for termination of the monitored named instance application; and
3 signaling termination of the monitored named instance application to a
4 cluster service.

1 25. A method according to Claim 23, further comprising:
2 checking for termination of the monitored named instance application; and
3 restarting the waiting subroutine.

1 26. A computer-readable storage medium holding code for performing
2 the method according to Claim 20.